Book of Kali:
Basics

Author: Seccult
Date: March 28’th 2026
Version: 1.0

Table of Contents:

Introduction: - pages 1-2

Linux Terminal Command Line Basics: - pages 3-9

Linux Permissions, Operators, Classes, Octals, and “chown” Command Overview: - pages 10-13

Linux Nano Command Line Text Editor: - pages 14

Linux Nano Text Editor (Tips and Tricks): - pages 15

Setting up Wifi (via terminal): - pages 16

Additional Fun and Useful Terminal Commands: - pages 17-18

Conclusion: - pages 19

Introduction:

whoami: So here is the deal, I am a multi certified pentester, and defense analyst, for years I've survived as a scrappy freelancer in the industry.

I very recently took a job offer that spoke to my heart, but prior to this position I turned down several six figure job offers because they sounded fucking retarded, and I think I woulda kissed the end of a shotgun had I taken them.

I'm a high school dropout, never went to no fancy university, my set of knowledge came from being self taught, and weaponizing my autism.

When I started my journey, it came out of a place of love for privacy and freedom, I wanted to understand how these systems worked, and that was enough to carry me through.

These guides/tutorials have been painstakingly converted from my original handwritten notes, and as I complete converting more I will upload them here free of charge.

I'm doing this because when I was starting out I had a difficult time getting help, and I got put down so much, some IRL people I knew actively discouraged me from even attempting to learn. Now I am in a position where I'm living my life doing the thing I love, for reasons I feel good about, and I reached something people spend their entire lives seeking, fulfilment.

If I listened to everyone telling me I couldn't, or shouldn't do something, I wouldn't be here, I said fuck em instead, and I did the thing a lot of people thought I couldn't do, it wasn't always easy, but I did it.

I'm writing all this out to give some context, and to say you can learn about these systems from zero, you don't need to go to university, or pay exorbitant amounts to do so, and if you genuinely want to learn these are my notes free of charge to help you.

Book of Kali Basics Overview:

This manual is meant for people starting their journey from **zero** practical linux experience, it's not a theortical course, so you should already know the lingo, for instance what a **terminal** is. (if there is enough demand I may do a theoretical course).

You should also already have access to the **kali OS** as a VM instance (or some other means), this won't include a tutorial on how to download, install, and run kali, you need to figure that out yourself.

Note: This tutorial was written with kali in mind, you can use another OS, and it should be okay, but you may have issues following certain sections if you do.

Say you've managed to do that? Well welcome, doing these courses (in order) will take you from linux virgin riding the internet bareback, to someone proficient enough to use protection, ready to do

a bit of penetration testing .

This guide is the first in a set of guides meant to teach one the fundamentals, if you do them all, you are going to get a foundation for linux usage, and defensive tools, before I release my next course on some offensive tools (again free of charge).

You won't become a 1337 hacker, but you will gain enough knowledge to continue learning, and operate a terminal comfortably, and move laterally from there.

There are going to be 2 guides in this series:

1). Basics.
2). Foundational Privacy.

One more thing:

Please feel free to offer some suggestions, but just be aware I'm releasing these guides for free, and I may not be able to make changes right away (or at all), sorry, I’m fucking crazy busy.

Note about copywrite: This document is free to read, download, share, just link to the official source, and please give me credit.

Linux Terminal Command Line Basics:

Overview: This section will discuss the most common terminal commands used for standard command line operations, as well as provide a description as to what these commands are used for:

cd This will change your current directory. For example typing **cd** followed by the **directory** you would like to access will take you to that directory location.

Command Format Example:

cd [directory location]

Practical Command Format Example:

cd /root/Desktop

The command format example listed above will take one to the **~/Desktop** directory location.

Note: Directory paths are **case sensitive**, one must make sure they input the correct lower, and uppercase characters to be taken to a directory.

Other cd commands:

cd .. Typing in **cd** followed by **..** will bring one to a previous directory. For instance if one is in the **/dev** directory location, then by typing in the **cd ..** This command will now take you to the **previous directory**, which in this case would be the **/** directory location.

ls This command will **list all files**, and **sub-directories** within one's **current directory**.

Note: ls followed by a specifed directory will give one the listings for that specific directory.

For instance, **ls /etc** will list files, and folders from the **/etc** directory, instead of your current directory.

Other "ls" commands:

ls -l This command works the same as the **ls** command, however the command output will include additional information such as:

- Who the owners are.
- The size of the files, and folders.
- Date, and time related information.
- Permissions.

ls -la This command will list **hidden directories**, as well as who the owners are, and what groups own a file.

cp This command will **copy a file**, and place it in a desired directory. In order for the file to be placed where desired, the location must be specified.

For instance, if one had a file called "**copytest**" in the **/root/Desktop** directory, and one wished to copy said file to a directory called "**bookofkali**"; the the command to do so would be as follows:

cp copytest /root/Desktop/bookofkali

Other "cp" commands:

cp -v The **-v** command stands for **verbose**, by including the **-v** in the **cp** command, one can get the output of the file they copied. In other words, **cp -v** will tell one what was copied; and to where.

Practical Command Format Example:

cp -v copytest /root/Desktop/bookofkali

The above command would provide the following output:

'copytest' -> '/root/Desktop/bookofkali/copytest'

Note: The **-v** command option stands for verbosity, and is an extremely important command to include as it helps one verify the command functioned as intended.

mv This command will move, as well can be used to change the name of a file if needed.

How to change a file name using the "mv" command:

To change a file name using the **mv** command, type **mv** followed by the file name you wish to change in the current directory you are in; followed by the new file name.

For instance if one was in the /root/Desktop directory, and they wished to change the name of a file in that directory called **copytest** to **newcopytest**; the command to do so would be as follows:

mv copytest newcopytest

This command will rename the **copytest** file to **newcopytest** within that directory.

How to move a file using the "mv" command:

To move a file using the **mv** command one must type **mv** followed by the filename, and finally the end of the path one would like to move their file to.

For instance if one were in the directory of the file they wished to move to another directory, the following practical command format example could be used to do so:

mv -v testfile /root/Desktop/bookofkali

In the above example, after executing the command, the file from the directory one is currently in will be moved to the **bookofkali** sub-directory, located within the **/root/Desktop** directory.

Note: If one were to wish to move a file from a directory they are not currently in, to a different directory, one could do so by specifying the path of the target file they wish to move, followed by the path where the file is to be moved.

Practical command format example:

mv -v /media/testthisfile/newtest /root/Desktop/bookofkali

The above command will move the newtest file from the **/media/testthisfile** directory, and will place the file in the specified **/root/Desktop/bookofkali** directory.

Other "mv" commands:

mv -v The base **mv** command will work in conjunction with the **-v** command, this will give one an output of the file that is being moved.

This is extremely important to include in ones command in order to understand whether the command succeeded, and where the targeted files were moved, or what they were renamed.

mv -i This command will prompt one if a file is at risk of being overwritten. For instance, if one were to move a file from one directory with an **identical name to another file** located in the directory one is moving it into, they will be given a prompt.

pwd This command stands for **print working directory**, or **present working directory**. Using the **pwd** command will let one know what directory they are currently in.

man This command stands for **manual**.

Using this command followed by another command name, will give one a full description of what the specified command does. In this way the **man** command is a fantastic way to get help, and understand all of what another command can do.

Practical command format example:

man mv

By using the above command one will receive a description of the **mv** command.

clear This command will clear the terminal of any text.

cat This command will give one a quick overview of a file, this is a safe way to view contents of a file, since it is only possible to view, and not edit a file using the **cat** command.

Practical command format example:

cat testfile.txt

The above command would allow one to look at what is inside the **testfile.txt** file, again because the **cat** command is being used one will not be able to edit, or make alterations to the file.

Note: The **cat** command will leave the overview of the file one selected within their current terminal history.

This is useful if one needs a reference within the terminal of the file one used the **cat** command on for later use.

less This command is very similar to the **cat** command in that it ostensibly performs the same function, i.e. gives one a overview of a file.

The **key difference** is that while the **cat** command will give one an overview of the file within the terminal session itself; the **less** command will give one an overview of the file within it's own window it creates within the terminal session.

To elaborate, this means when one closes the window that the **less** command creates to view the file, the associated overview of that file will **NOT** remain in one's terminal history.

This is a useful alternative to the **cat** command for privacy, and to keep one's terminal neat.

Command format example:

less [desired file]

Practical command format example:

less testfile.txt

The above command will give one an overview of the testfile.txt.

grep This command provides a way to filter, as well as search within specified files using delineated critera (usually words).

For **grep** to be properly used, it needs to be "**piped**" to the specfic file one wishes to search, or filter, and can be used in conjunction with the **cat** command.

Note: This **|** character is used to pipe something, it is a symbol, and **not** to be confused with a capital i, or lowercase L.

Practical command format example:

cat torsocks.conf | grep path

The above command will search the **torsocks.conf** file (this file is located in the **/etc** directory) for the word **"path"** using the **grep** command.

As one can observe a **pipe** is used to link, or "connect" the **grep** command to the specifed **torsocks.conf** file. The search, or filter term (in this instance tge word path) needs to be specified after the **grep** command.

echo This command will literally echo whatever one types into the terminal. To properly echo what one would like, one **MUST use quotations.**

Practical command format examples:

echo "I hack time"

By using the above command one will echo the words **I havk time**, the reason these words get echoed is because they are encased between the **quotation symbols ""**.

Note: One may also use the echo command to insert, and change the text, or values of a file which one specifies in their command.

Practical command format example:

echo "I hack time" > testfile.txt

The above command will echo the words that are encased in between the quotations to the file **testfile.txt**

In order for one to propery target a file to echo to, a **greater symbol** must be used after whatever one has selected to be in the quotation marks; with the file you would like to target specified after the greater symbol **>**.

touch This command is a quick method to create one, or several files within the directory one is in, or a directory they specify.

Practical Command Format Example:

touch /hellofriend/secrets.txt

The above command will create a new text file called **secrets.txt** in the **/hellofriend** directory that was specified. The file one wishes to create must be written at the end of the directory path in which it will be created.

Note: If one is already in the directory where they desire the file to be created, they simply need to use the **touch** command, followed by they file they wish to create. In this instance, since one is already in the desired directory, no directory path is required.

Practical Command Format Example:

touch topsecret.txt

The above command will create the file **topsecret.txt** in the directory one is presently in.

mkdir This command will create a directory (or "folder") within whatever directory one is currently in, or within a directory that one specifies.

Practical Command Format Example:

mkdir /hellofriend/FKAsongs

The above command will create a sub directory called "**FKAsongs"** within the **/hellofriend** directory specified. The sub directory one wishes to create must be written at the end of the directory path in which it will be created.

Note: If one is already in the directory where they desire a **sub directory** to be created, one simply needs to use the **mkdir** command followed by the sub directory they wish to create. In this instance, since one is already in the desired directory, no directory path is needed.

Practical Command Format Example:

mkdir whereismymind

The above command will create the directory **whereismymind** in the directory one is presently in.

rm This command will delete a file, or directory, Once one removes a file (or directory) with this command, the file will be next to impossible to recover. **Use rm with extreme caution!**

Note: As a failsafe **rm** on its own will not be able to remove directories.

Other "rm" commands:

rm -r This command will remove directories, and their contents recursively.

rm -i This command will prompt one before every removal. It is good practice to include this command option as an extra prevention when deleting files, or directories.

rm -v This command will provide verbosity output, that can be used to understand what has occured once the command has been executed. Specifically it will show one what files, or directories have been deleted.

locate This command will search for files by name.

Command format example:

locate [file name]

Practical command format example:

locate gubgubsnailman

The above command will search ones computer for any files with either exact, or similar names to **gubgubsnailman**.

Other "locate" commands:

locate -b This command will match **only** the base name against the specified patterns. This command option works the opposite of **–wholename**.

Linux Permissions, Operators, Classes, Octals, and “chown” Command Overview:

Permissions within linux systems are given to files, and are used to allow, or deny access to different users, or groups. Respectively there are **three** permissions integral to understand in order to **read**, **execute**, or make a file **writeable.**

Permissions List:

r This file permission stands for **read**, the read or **r** permission in conjunction with a **+**, or **-** operator determines whether a file can be read or not.

x This file permission stands for **execute**, the execute or **x** permission in conjunction with a **+**, or **-** operator determines whether a file can be executed or not.

w This file permission stands for **write**, the write or **w** permission in conjunction with a **+**, or **-** operator determines whether a file will be writeable or not.

There are several operators one can use to add, remove, or allow the class exactly the permissions, no more, no less.

Operator List:

+ add permission.

- remove permission.

= allow the class exactly the permissions.

Permissions one wishes to add, or remove must follow, and be used in conjunction with one of the operators list above.

For instance **+r** would **add** a **read** permission to a file.

Note: Along with **operators**, **permissions** can also be used with **classes**, in the next section classes will be discussed. However unlike operators, one does not need to add a class in order to change permissions.

Classes Overview:

Classes may be used in conjunction with permissions, and its operators to change a permission of a file for a **User (owner)**, the **group**, the **others**, or all of the above.

Classes must come before and be used in conjunction with an operator, and permission.

Command format example:

chmod u+x [file]

This command will add an execute permission for the owner of the specified file.

Classes List:

u This class stands for **Owner/User**.

g This class stands for **Group**.

o This class stands for **Others**.

a This class stands for **ALL**.

The "chown" command overview:

chown This command allows one to change the user ownership of a file.

For example if the owner of a file was not **root**, but one wanted to change it to **root**, one could use the following command format example:

chown root:root test

To illustrate in the above example, after **chown**, one would type in the **user name** followed by a **colon**, then the **user group**, lastly followed by the **file name**. This will effectively change ownership of the file **test** to the username **root** from the user group **root**.

Note: The username, and the user group are usually the same on a home **p.c.**, unless you are on a bigger server.

chmod This command allows one to change file permissions, when used with an operator, followed by the filename of the specified file who's permission one wishes to change.

Practical Command Format Examples:

chmod +r [file]

This command will add a **read** permission to a file.

chmod -x [file]

This command will remove a **execute** permission from a file.

chmod usage in conjunction with classes:

Along with operators **chmod** can also be used with **classes** to change a file permission.

Practical Command Format Examples:

chmod a+x [file]

Adds execute permission for all users.

chmod a-x [file]

Removes execute permission for all users.

chmod g-x [file]

Removes execute permission for the files group.

chmod a=x [file]

Gives all users execute permission, nothing more nothing less.

chmod o=rx [file]

Sets other users permissions to read and execute, but not write.

chmod ug+x [file]

Adds execute permission for the owner and the group.

Octal Bitmask Overview:

The other syntax that can be used with **chmod** involves an **octal bitmask** to represent each possible set of permissions, this is a less intuitive method for adding permissions, but wortb understanding.

In the octal representation, a **three-digit** number is used to express the permissions on a file.

Each digit is the **sum of three bits**, with the following numeric values representing the following permissions:

4 **read** permissions
2 **write** permissions
1 **execute** permissions

These bits can then be **added together** to give a **unique total**, as demonstrated in the examples below:

6 This is a combination of the **4** **read** bit, plus the **2** **write** bit, and allows for both permissions.
5 This is a combination of the **4** **read** bit, plus the **1** **execute** bit, and allows for both petmissions.
7 This is a combination of **all** the **three** bits, and provides **full** permissions.

The positioning of the bits in an octal bitmask determine whether the pemission is set for the owner, followed by the group, and lastly others.

To illustrate further, **755** is the same as **rwx-rx-rx**

Note: This syntax can be used interchangeably with the other in **chmod**.

Octal Bitmask Examples:

chmod 111 [file]

This command gives all users execute permissions, nothing more nothing less.

chmod 754 [file]

This command gives the owner full permissions, the group read and execute permissions, and the others only read permissions.

Linux Nano Command Line Text Editor:

The **nano** text editor is a pre-installed program to edit.txt files from the terminal.

In order for one to edit a file using the nano text editor, they must type in **nano** into the terminal; followed by the file (or path to specified file).

Command Format Example:

nano [file]

Practical Command Format Example:

nano root/Desktop/testfile.txt

The above command will allow the nano text editor to open the **textfile.txt**, located in the **root/Desktop** path that was specified.

Using the nano text editor usage guide:

Once one is in the nano text editor, it is extremely important to know that one must use the **Ctrl** key in conjunction with another of the specified keys at the bottom of the nano footer, in order to execute a desired action.

For example, if one wanted to exit the program, one would press the following key combinations **Ctrl** and **x** at the same time.

Additional examples:

^G is the **Ctrl+G** keys.
^O is the **Ctrl+O** keys.
^W is the **Ctrl+W** keys.

Note: As one can observe the **^** symbol represents the **Ctrl** key.

Getting help within nano:

Within nano one can familiarize themselves with it's functions, as well as get help by pressing **Ctrl+G**.

This will give one an overview of the nano functions, and what they do.

Linux Nano Text Editor (Tips and Tricks):

Copying, Cutting, and Pasting:

When one wants to copy text within a graphical environment (standard desktop application), one needs to first highlight it with the cursor. Similarily, within nano one "marks" text by using the **Ctrl+^** command.

To do so one simply moves the cursor to where they want to start marking, and proceed to "set" it by pressing **Ctrl+^**, this will "mark" everything between the starting point up to, and **NOT** including the cursor.

Note: When using the **Ctrl+^** command the **^** key can be found on the **6** key at the top of most keyboard layouts.

Other Notes Regarding Copying:

- Copying/Cutting will not include empty space.

- One can also mark backwards from their set point, one must take caution when doing so as they can edit text while marking!

Note: If one makes a mistake they can press **Ctrl+^** again to **unset** the marker and start over.

Other Copy Related Command Options:

Meta+^ This allows one to copy marked text.

Note: Meta respresents the **esc** key, or on some computers the **alt** key. As such to get the above command to work, one should actually press the **esc+^** keys (or in some cases **esc+alt).**

Ctrl+K This allows one to cut marked text.

Ctrl+U This allows one to paste their text once one their cursor to a suitable position.

Ctrl+W This allows one to search for a text string.

Setting up Wifi (via terminal):

Overview: This section will give a brief overview of how one may set up a wifi connection using a network interface card via their terminal.

1). Open a terminal session, then within the terminal input the following command:

ifconfig

This command will show you the different interfaces available from ones device, as well as the status of each interface.

The 3 standard interfaces are as follows:

- eth0
- lo
- wlan0

Note: Using the **ifconfig -a** command will display complete listings of all the interfaces, including those that are “down”.

2). These interfaces can be turned on, and off by using the **up**, and **down** commands in the terminal. This is useful to test one's wifi connection, as well as to enable, and disable it via a specifed interface.

Enabling a network interface card command format example:

ifconfig [desired interface] up

Enabling a network interface card practical command format example:

ifconfig wlan0 up

Disabling a network interface card command format example:

ifconfig [desired interface] down

Disabling a network interface card practical command format example:

ifconfig wlan0 down

Additional Fun and Useful Terminal Commands:

The following commands are good for a new user to learn, as they allow one to familiarize themselves with how to operate a terminal to perform actions they would normally be accustom to doing from a desktop GUI environment.

This section also includes a final task to download and use the **cmatrix** tool, a fun terminal application to help keep one motivated as they proceed in thier journey to become a defensive, or offensive security operator.

Useful supplementary commands:

cal This command will present a calander within ones terminal session.

reboot This command will restart ones computer.

shutdown -h now This command will shutdown ones computer immediately.

Ctrl+z This command will stop a terminal command when pressed.

date This command will bring up the date within the terminal.

rev After typing this command into the terminal, press **enter**, now typing in a word and pressing the **enter** key will cause the word you just entered to be reversed.

firefox This command will start a firefox session from your terminal.

Capstone challenge: downloading "cmatrix" from the terminal:

cmatrix is a fun graphical tool that will change ones terminal session into a screen similar to the one in the matrix films. This tutorial gives one a guide on how to download the tool, and the previous commands from the prior sections in this manual can be used to determine how to use it, one is encouraged to explore the tool using the knowledge learned prior.

How to download cmatrix tutorial:

1). Start a new terminal session.

2). Within this terminal session one may download cmatrix using the following command:

apt-get install cmatrix

3). Once downloaded, one may execute the program using the following command:

cmatrix

If successful the screen effect from the matrix will run in your terminal session.

Note: To stop the **cmatrix** program from running, simply press the **q** key.

Conclusion:

This concludes this guide, the next guide digs deeper into linux operations, and how one can harden the privacy on thier system. I may add screenshots, and additional content at a later date, if I do I'll add a revision section near the start of this document.

Thank you for reading, well done on making it to the end! Please follow, and like this post if you want to get updates as I release more guides, and infosec related content.